(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
15 February 2001 (15.02.2001)

PCT

(10) International Publication Number
WO 01/11516 A2

(51) International Patent Classification[7]: G06F 17/60

(21) International Application Number: PCT/US00/20765

(22) International Filing Date: 28 July 2000 (28.07.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

| | | |
|---|---|---|
| 60/147,130 | 4 August 1999 (04.08.1999) | US |
| 09/375,865 | 17 August 1999 (17.08.1999) | US |

(71) Applicant: **SOLBRIGHT INCORPORATED** [US/US]; 7th Floor, 212 Fifth Avenue, New York, NY 10010 (US).

(72) Inventors: **COMPTON, Anthony, K.**; One Irving Place, #V-11-I, New York, NY 10003 (US). **HABBOUB, Fouad, K.**; 55 Pineapple Street, Brooklyn, NY 11201 (US).

(74) Agent: **KALOW, David**; Kalow & Springut LLP, 488 Madison, 19th Floor, New York, NY 10022 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

**Published:**

— *Without international search report and to be republished upon receipt of that report.*

*For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.*

WO 01/11516 A2

(54) Title: METHOD AND APPARATUS FOR PROCESSING AND REVIEWING ADVERTISEMENT SUBMISSIONS FOR CONTENT AND CONFORMITY

(57) Abstract: An advertisement management system for receiving, processing and reviewing advertisements for content and conformity for embedding into a Web page published over the Internet. Advertisers or advertising agencies send advertisements to the advertisement management system for processing. The processing includes modifying the submitted advertisement and comparing it to Web site guidelines before displaying the advertisement for visual inspection. Once approved, the advertisement is submitted for publication on a Web page over the Internet.

# METHOD AND APPARATUS FOR PROCESSING AND REVIEWING ADVERTISEMENT SUBMISSIONS FOR CONTENT AND CONFORMITY

## CROSS REFERENCE TO RELATED APPLICATIONS

This application claims the benefit of United States provisional patent application No. 60/147130, filed August 4, 1999, the entire disclosure of which is hereby incorporated herein by reference.

## FIELD OF THE INVENTION

The present invention broadly concerns publication of advertisements on Web pages and more particularly concerns a system and method for receiving, processing, and reviewing advertisements for content and conformity for embedding in a Web page published over the Internet.

## BACKGROUND OF THE INVENTION

Web pages frequently include embedded advertisements. Different Web pages have different requirements for placement of an advertisement on the page. For example, the requirements for placement of an advertisement on a Web page may vary in file type, file size, number of characters, number of images, and Java, or JavaScript facilities. An advertiser or advertising agency may produce a single advertisement which the advertiser or agency would like to publish in many different web pages. However, such an advertisement might be compatible with some web pages and not others. For example, the advertisement might be 4000 characters in size and the Web page advertisement space may only be able to accommodate 3000

characters in size. This discrepancy in size would result in the advertisement not being properly displayed, and the advertiser or agency would not be aware of the problem until the advertisement is actually published and viewed by the advertiser or agency.

To avoid such problems, certain tasks are implemented manually by production facilities for Web pages. These tasks include having a person review the advertisement before it is published on a Web page to verify that the advertisement meets the requirements of the page. The person may receive advertisements on a disk sent by mail or may receive advertisements through e-mail delivery. Upon receiving the advertisements, the person would have to load the files into a computer system and identify and remove all necessary advertisement attachments and make copies of each. The person would also have to manually calculate the file size, number of lines, number of characters, number of images, check for Java and JavaScript programming code, and remove unnecessary or incompatible code or add necessary code. The person would then have to manually compare the advertisement properties to the particular Web site parameters in which the advertiser or agency wants to publish the advertisement. Manual application of these tasks can result in errors and reduced capacity for hosting advertisements. The tasks involved in processing advertisements for conformance to a particular web site is both tedious and time consuming.

Accordingly, there is a need for a system that automates the process of receiving, processing, and reviewing advertisements for content and conformity for embedding in a Web page published over the Internet.

## SUMMARY OF THE INVENTION

The present invention satisfies the above described needs by providing a method and a system for receiving, processing and reviewing advertisements for content and conformity to requirements for embedding in a Web page published over the Internet. The method includes receiving a submission from an advertiser or advertising agency. The submission may contain multiple file attachments, html code, and information about where to locate a creative asset on the World Wide Web. The submission is searched for creative assets by the advertisement management system. The creative assets may be text code, image code, and rich media code that are processed to form and advertisement. Rich media is a media type such as JPEG, GIF/animated GIF, Enliven, InterVU, Comet Cursors, VRML, Shockwave, and Flash. The creative assets are modified so that they are in a form that is appropriate for publication on an existing Web page, and are compared to Web site guidelines for conformity to the guidelines as part of an approval process.

The advertisement management system includes a front-end system for providing the user with the ability to control the system. The front-end system uses a standard web browser as a user interface to the advertisement

management system. A mail system receives, transmits, and processes submissions sent to it by an advertiser or an advertising agency. A business system coupled to the front-end system and the mail system modifies the creative asset and compares it to Web site guidelines for conformity as part of the approval process.

## BRIEF DESCRIPTION OF THE DRAWINGS

FIG 1. is a block diagram of a preferred advertisement management system.

FIG 2. is a flowchart outlining the general assembly process of a rich media advertisement.

FIG 3. is a flowchart illustrating in greater detail steps 62 through 72 of FIG 2 according to one embodiment of the present invention.

FIG 4. is a flowchart illustrating a front-end pickup URL process.

FIG 5 is a flowchart illustrating a back-end pickup URL process.

## DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

A preferred embodiment of the present invention is directed to an advertisement management system for receiving, processing, and reviewing advertisements for content and conformity for publication on the World Wide

Web. The advertisement management system automates the process of receiving, reviewing and processing creative assets for publication on Web pages. Submissions are sent by advertisers or advertising agencies through e-mail to the advertisement management system for processing. The submissions include creative assets, which are elements that will be processed and combined to form a Web page advertisement. The submissions may also contain messages that include a date, a sender's name and attachments or images. Additionally, the submission may include multiple file attachments, html code, or information about where to locate an advertisement on the World Wide Web. When a submission is received by the advertisement management system, it is separated into components, where the components may be multiple file attachments (creative assets), text, etc. The advertisement management system processes the creative assets and modifies them, if necessary, and compares them to Web page guidelines for conformity. The preferred advertisement management system provides for the creation of filters and ad models which can serve as Web page guidelines. The system provides the ability to review new advertisement submission for content and for conformity to the ad models.

A block diagram of an advertisement management system formed in accordance with a preferred embodiment of the present invention is illustrated in figure 1. The advertisement management system preferably includes a front-end system 2, a back-end system 28, a database 6, an adapter system 8, and an ad server database 10. The front-end system 2 is coupled to the

back-end system 28 and the database 6. The front-end system 2 provides a user interface to the advertisement management system. The back-end system 28 receives submissions and processes them into advertisements and stores them in the database 6. Data for the advertisement management system is stored in the database 6. The adapter system 8 is coupled to the database 6 and ensures that data stored in the database 6 is transferred to the ad server database 10, which stores the advertisements for use by an ad server.

Preferred embodiments of the present invention facilitates preparing rich media advertisements for use on web pages. The front-end system 2 preferably includes at least one web browser 12, a http web server 14, and Java servlets 16. The web browser 12 provides for user access to the advertisement management system. There are several commercially available web browsers, however, the preferred web browser 12 is Netscape Navigator from Netscape Communications. The web browser 12 provides the user with a window and various controls through which data from the web server 14 can be viewed and navigated. The user selects advertisement management system features and options via the web browser 12. Using the web browser 12, the user may enable different filters to apply to a selected creative asset and may review the creative asset for content and attributes.

The http web server 14 provides the hardware for communication between the web browser 12 and the back-end system 28. The user

commands executed at the web browser 12 are transferred through the http web server 14. The http web server 14 transmits and receives data between the web browser and the back-end system 28 and the database 6. The data provided by the web server 14 will be in the form of pages of data that can be examined using typical browser software. The web server 14 preferably includes Java servlets 16, which in the representative embodiment is the common gateway interface (CGI) program. The Java servlets 16 are software objects that control the retrieval and presentation of data onto the web browser 12, from the database 6, and which communicates with the back-end system 28. The CGI program is a program interface between the web server 14 and the back-end system 28 and the database 6. The CGI program lets the back-end system 28 process html forms and other data coming form the web browser 12, and then lets the back-end system 28 send responses back to the web server 14, which is delivered to the web browser 12. A main purpose of this program is to provide web-to-database access for the html interface. There is also logic in the program to process the commands selected by the user (i.e. approving banners, sending rejection e-mail). HTML templates served up by the CGI program are the main interface to the back-end system 28 and the database 6. Generally, the CGI program fills out the html templates and returns them to the web browser 12. In particular, the CGI program selects e-mail messages from an e-mail table in the database 6, and displays them to the user.

The back-end system 28 preferably includes three systems. each of which functions as five daemons and together function as a multithreaded application running in the background as a daemon process. The e-mail system 4 functions as a post office protocol (pop) checker daemon, a simple mail transfer protocol (smtp) checker daemon and an e-mail sender daemon . The ISD/Validator system 26, functions as an infoscanner daemon (ISD) and a validator daemon (validator). Each of these daemon functions run as an independent thread of execution within the daemon process. All five threads may be running within the daemon process at the same time and any of the threads may be enabled or disabled from the user screen of the advertisement management system interface. The back-end system 28 receives e-mail from agencies and/or advertisers through the simple mail transfer protocol (smtp) or post office protocol (pop3) e-mail system 4. The smtp is a TCP/IP standard protocol for sending e-mail between servers. The pop is a standard protocol that provides for client mail-reading programs to retrieve messages from a mail server. The e-mail system 4, includes the pop checker daemon which polls remote pop3 accounts for incoming e-mail submissions. The smtp checker daemon polls local smtp accounts and folders for incoming e-mail and downloads them into local directories in the database 6. The e-mail system 4 also includes an e-mail sender daemon which sends out e-mail to advertisers with assets that are rejected. The ISD/validator system 26, includes the ISD daemon which parses downloaded e-mail messages for assets and scans each asset for various properties, and matches the asset against user defined and industry standard configured

placement groups. The ISD/validator system 26 also includes the validator daemon which synchronizes the assets that are approved by the user and sends them to an ad server.

More specifically, the back-end system 28 includes the pop checker daemon that is responsible for retrieving e-mail from a pop server, located outside the advertisement management system. The pop checker logs into the e-mail account server and retrieves any new e-mail that has come into the e-mail account server. The pop checker, utilizing e-mail processors 20, extracts any zip files into individual files and reattaches the individual components to the e-mail. The e-mail and the attachments are stored in the database 6 file system under specific reference tables. After saving the e-mail and the components, the pop checker pings (Packet Internet Groper) the ISD/validator 26 for further processing. The Packet Internet Groper is a program used to test reachability of a destination by sending it an echo request and waiting for a reply. The smtp checker daemon is responsible for handling the e-mail that comes in by way of the smtp accounts. The smtp checker performs the same functions that the pop checker performs, with the exception that the smtp checker includes the ability to log into smtp compliant mail boxes, which are mainly used for internal system e-mail. The main responsibilities of the pop checker daemon and the smtp checker daemon are to receive all e-mail sent to the advertisement management system and to extract all of the attachments that come in with the e-mail. The e-mail is separated into components, where the components may be multiple file

attachments that are advertisements (creative assets), html code, or text that includes information about where to find the creative asset on the Web. The text may include a universal resource locator (URL) of which the user can use the web browse 12 to access the particular Web page and obtain advertisements posted on the page. The attachments are put into a list and an attachment list is created along with an instruction list. The e-mail sender daemon, located within the e-mail system 4, is responsible for sending e-mail to the outside world from the advertisement management system.

Preferably, the business system 22 includes Java objects 24 and the ISD/Validator 26, which includes the infoscanner daemon (ISD) and a validator daemon. One process performed by the Java objects 24 is a work queue. The work queue identifies a status of each submission, such as new, processed, deleted, other, and permits the user to view other attributes of the submission. The work queue may sort by subject or date and may select a single group or all submissions for forwarding or deleting. The work queue may also forward the submission for processing and may forward submissions for group processing. The Java objects 24 also controls the submission process of advertisements to the ISD/validator 26. This interface allows the user to process every aspect of the submission by providing a display screen that allows the user to select various options. The screen that is displayed contains a formatted view of the e-mail, options for searching for other submissions and advertisements, the ability to add an html asset (html submission), addition of a text asset (text submission), or pick-up an asset

from a web location (pick-up URL), select an advertiser and/or campaign, process each asset (create/edit, reject, ignore, duplicate), and process the advertisement.

The infoscanner daemon (ISD), located within the ISD/Validator system 26, is the main engine of the back-end system 28. The ISD reads the e-mail that enters into the system through pop checker 18 or smtp checker 18. The e-mail is parsed and different types of ads are constructed out of the submission. Each ad is validated against a set of rules and the result is stored in the database 6. The ISD applies passive validation rules (ad models) and applies active rules (filters) that modify the html ads that come into the system. The active and passive rules can be configured through the front-end system 2 and stored in the database 6. After the ISD has completed the processing of the advertisements, the advertisements become available for user interaction.

The validator daemon, located within the ISD/Validator system 26 handles submissions from the front-end system 2, unlike the ISD that handles submissions that come in through the e-mail system 4. The front-end system 2, through the pick-up URL functionality, picks up ads from a web page and submits it to the validator. The information transferred to the validator from the front-end system 2, is conveyed through a ping-string that informs the validator as to the type of submission and the location of the asset retrieved by the pick-up URL process. The validator applies the ad

models and filter rules to the advertisements that come in by the front-end system 2. The ISD applies the ad models and filter rules to the advertisements that come in by the e-mail system 4.

The database 6 is coupled to the back-end 28 and makes use of standard database technology to manage creative assets, campaigns, and account information. The database 6 stores modifiable records with information about the e-mail accounts used for creative submissions, submissions and client correspondence, ad models, filters, and templates used to preprocess creative asset, system user information, and advertisements, agencies, advertisers, and campaigns. Data definition and manipulation is optimized through the use of the Structured Query Language (SQL).

The adapter 8 is the interface between the database 6 and the ad server database 10. Any changes in the advertisement management system, such as an ad being approved and submitted, are synchronized with the ad server database 10 by the adapter 8. The adapter 8 consists of three systems: an extract system, which pulls the changed data from the database 6; a map or transform system, which changes the format of the data from the advertisement management system data model to the ad server data model; and a store system, which inserts, updates or deletes the mapped data on the ad server. These functions ensure that the ad server database 10 is updated with any new information generated by the advertisement

management system.

Figure 2 illustrates the general flowchart for processing rich media code by the ISD/Validator system. Preliminary to the steps in figure 3, the smtp/pop e-mail system and the e-mail processors have communicated to the ISD/Validator that e-mail has been received and is available for processing.

Generally, at step 50, the ISD/Validator searches the e-mail for a rich media file (e.g. html code). Rich media is a media type such as JPEG, GIF/animated GIF, Enliven, InterVU, Comet Cursors, VRML, Shockwave, and Flash. If a rich media file is found, then at step 54 a rich media advertisement is created. At step 56, all creative assets (e.g. rich media) referred to within the html file are retrieved. The creative assets are elements that will be processed and combined to form an advertisement. The elements may include text, images, or rich media code. If there are references to assets within the html file, then the assets are saved to a database and under a specific directory, steps 58 and 60. If there are no assets present within the html file, then the html file bypasses step 60 and goes directly to step 62. At step 62, the html (e.g. rich media) filter parameters are retrieved from the database. The filter parameters represent selections made by the user at the front-end of the system. If there are any html filters enabled, step 64, then the html file is stored in its original format as a backup file, step 66. At step 68, the html filters that are enabled are constructed so that they may be applied to the html file. The html file is scanned through the enabled filters, step 70, and

the resulting filtered html file is stored in the database, step 72. If there are no enabled filters, step 64, then steps 66, 68, 70, and 72 are bypassed. At step 74, the properties (e.g. file size) of the html file are retrieved by scanning the file and storing the results in the database. At step 76, ad model parameters are retrieved from the database. The ad model parameters represent Web site guidelines for displaying advertisements. In step 80, if ad models are found within the database, step 78, then the properties retrieved in step 74 are compared to ad model parameters retrieved in step 76. The ad model parameters that match the html properties are identified and stored in the database, step 82. If no ad models are located, in step 78, then steps 80 and 82 are bypassed. At step 84, the status of the html file is changed to "new" and the file is ready for processing by the front-end of the system.

More specifically, at step 50 of Figure 2, the ISD/Validator system searches the e-mail for any attachments that contain files with html extensions. If html files are not found, then the html processing ends and processing for image files, text files, and unknown files begin, step 52. If html files are found, then an html advertisement is created, step 54. Creating an html advertisement, step 54, involves creating a record for the html file in the database. The record is given a unique identification code and any references to the html file are by its identification code. Additionally, any files related to the html file will also be stored in reference to the identification code.

At step 56, the html file is parsed for all creative assets referred to within the html file. The creative assets are elements that will be processed and combined to form the advertisement. The elements may include text, images, or rich media code. The ISD/Validator system parses the html file by reviewing the html file for any assets located within the html file. The ISD/Validator system searches for anything that refers to another file that would become an element in the display of an html screen. This is achieved by searching the html file for tags that reference image files. An example of the types of tags that the ISD/Validator system searches for are "Source =" and "code =" tags. Attached to the tag is a file name, under which the image file can be located. The different types of html tags that the ISD/ Validator system searches for are GIF, JPEG, AU, AUI, Class files, and Shockwave files. Next, the ISD/Validator system searches all the attachments within the e-mail for the assets identified during the parsing of the html file. This is achieved by comparing the list of file names of the assets (found during parsing) to the attachments that are part of the e-mail. If assets are found, then they are removed from the e-mail attachment list and stored in the database under a specific directory. If there are references to assets within the html file, step 58, then the assets are stored in the database and moved to a specific directory, step 60.

Once step 56 is completed and the number of referenced assets are stored, the assets are then grouped. The assets, referred to within the html file, are grouped into image assets, text assets, and/or unknown assets. The

names of these assets are marked as being part of the advertisement and stored in the database under the appropriate heading. If there are no references to assets within the html file, step 58, then step 60 is bypassed.

At step 62 of Figure 2, the ISD/Validator system retrieves html (rich media) filter parameter names and values from the database and stores them in program memory. The filter parameter names and values reflect filter selections made at the front-end of the system by the user. Different filter parameter names and values make up combination filters. In order that the filters reflect the current status of enabled and disabled filter selections made at the front-end of the system, the filter parameters are retrieved from the database each time they are needed. If any html (rich media) filters are enabled, then the original html file is stored as a backup file in the database, steps 64 and 66. In step 68, the html filter is constructed. This requires retrieving atomic filters from the database and combining them to form the combination filters that were enabled in the front-end of the system.

At step 70 of Figure 2, the html file is applied to each of the combination filters that are defined in the front-end of the system. This is achieved by scanning through the html file character-by-character and/or line-by-line and comparing the characters and/or lines to the filters. If there is a match, then the filter is applied. The result is a new (modified) html file with all of the html filters applied, which may include insertions in the html file and/or deletions from the html file. At step 70, the new html file, is stored in the

database where it can be retrieved for further processing. If there are no html (rich media) filters enabled, step 64, then steps 66, 68, 70, and 72 are bypassed and the process continues with the original html file.

In step 74, properties of the new html file are determined and stored in the database. The properties are determined by scanning the new html file for file size, total number of lines, total number of characters, total number referred images in the html, list of all files referred in the html, total file size of all image files referred in the html, and existence of any references to Java or JavaScript.

In step 76, ad model parameters are retrieved from the database and stored in memory. The ad model parameters represent Web site guidelines for displaying advertisements. Because the parameters are set by the user at the front-end of the system and may be changed, the parameters are retrieved from a common database to reflect the current selection of the user and thus reflect the current ad model parameters. As a result, every time this function is performed, the rules are retrieved from the database in case some of the rules have been changed. The ad models are predetermined parameters that reflect particular standards or rules for placing an ad on a Web page. These rules include, but are not limited to, maximum file size, maximum lines, maximum characters, maximum images, allow Java, and allow JavaScript. The ad model parameters are set by the user at the front-end of the system.

If there are ad models in the database, step 78 of Figure 2, then the properties of the new html file are compared to all of the ad model parameters, step 80. If the new html file properties do not match any ad model parameters, then a warning string is constructed and stored in the database, step 80, and the warning string is noted as a warning status at the front-end of the system for the user to identify. If the new html file properties match one or more ad models, then an acceptance string is constructed and an acceptance status is noted at the front-end of the system for the user to identify and the acceptance string is stored in the database, step 80. With the status of the html file (advertisement) changed to new, step 84, the advertisement is ready for processing in the front-end of the system. The advertisement is displayed at the front-end of the system for visual inspection by the user. Once inspected, the user can assign the advertisement to a campaign and an advertiser.

The present invention provides the advertisement management system user the ability to select and create rich media filters to modify the rich media code in preparation for display on a web site. The user has the option to initiate a number of filters that may insert, delete, or update portions of the code to allow for proper display of the advertisement over the Web. There are two types of filters used in this system, standard filters and user defined filters. Standard filters are used to remove html header tags. Examples of standard filters include, but are not limited to: remove <head> tag, remove <html> tag, remove <meta> tag, remove <body> tag. These tags are part of a standard

web page and are also included in the creative assets submitted by the advertiser or advertising agency to the advertisement management system. However, because the creative asset (html file) is being placed within a web page that already includes the standard tags, the tags associated with the creative assets are not needed and are removed by the filtering process.

Another type of standard filter is an insert redirect string. The insert redirect string filter inserts a redirect string everywhere there is a <href> tag in the html file. Normally, when a creative asset (html file) is received from an agency it includes a URL link within the html file. The URL link is used to link a person who selects the advertisement to the web page of the advertiser (e.g. "clicking" on the advertisement to receive additional information). In order to track and count the number of viewers who actually go to an advertiser's web page, a redirect string is inserted into the html file that redirects the viewer to a tracking program, before actually going to the advertiser's web page. The tracking program tracks and counts the number of viewers who actually go through the advertisement to the advertiser's web page. Applying the insert redirect string filter results in a new URL (within the html file) that takes the viewer to a tracking program before taking the viewer to the original URL.

User defined filters are fitters that the user creates by defining certain available parameters. The parameters that are available in this system are various types of delete and insert parameters. "Delete Tags Only" is a user

defined filter where the user enters start and end tags, such as <a> and </a>, to be removed from the code, leaving the text between the tags in the file. "Delete Tags And Text Between Tags" allows the user to enter start and end tags that identify a section of code to be removed from a file. The text between the tags, as well as the tags, are deleted. "Delete Text" permits the user to select text to be deleted from a file. "Insert Text After" allows the user to enter text that is used to flag the insertion point for text to be entered into the file, and the new text is entered after the text flag. "Insert Text Before" performs a similar function as the "Insert Text After" function, except that the new text is inserted before the text flag. "Replace Text" allows the user to enter text to be searched for within the file and then replaces the text with user selected text. "Replace Text Between Tags" permit the user to enter start and end tags that identify a section of code to be searched, as well as the text to be searched and the replacement text. "Replace Text Between Tags And Tags" offers the user a similar option as "Replace Text Between Tags" except both the searched for text and the tags are replaced with the new text. Numerous search and replace operations can be created from the above listed parameter names.

The user configures the filters at the front-end of the system by enabling and disabling the filters of choice and then applying them to the html files. After the filters are defined in the front-end, they are applied to the back-end of the system where they are stored in the database. The back-end searches through the database to determine what filters are defined (enabled)

in the front end of the system. It then applies the filters to all of the html files that were stored in the database.

Figure 3 illustrates in more detail the process of constructing rich media (html) filters. At step 90, all enabled filters, standard and user defined, are retrieved. If there are no enabled filters, then the filtering process stops, step 92. If there is at least one enabled filter, step 92, then every atomic filter type related to the enabled filters are retrieved from the database, step 94, and stored in memory. At step 96, all parameters relating to each atomic filter are fetched from the database and stored in memory. At step 98, the atomic filters are then constructed. At step 100, combination filters are constructed out of the atomic filters. If there are more combination filters, step 101, then go back to step 96 and fetch all parameters relating to each atomic filter. If there are no more combination filters, step 101, then construct a filter chain, step 102. At step 104, the html file is retrieved from the database and stored in memory. At step 106, each combination filter is sequentially applied to the html file. If no errors are detected during step 106, then the results of step 106 are stored in the database. If an error is detected then the filtering process stops and a error string is created.

More specifically, at step 90, the ISD/Validator system retrieves all enabled standard and predefined filter parameters from the database and stores them in memory. If there are no enabled filters, then the filtering process ends, step 92. If there is at least one enabled filter, step 92, then

each atomic filter type that is related to the enabled filters is retrieved and store in memory, step 94. The combination filters are made of atomic filters, and the combination filters represent the filters that the user selects or creates at the front-end of the system. Examples of atomic filters are insert filter, replace filter, delete filter, and undelete filter. These filters can be combined in various ways to produce different combination filters.

At step 96, the parameters for each atomic filter are fetched from the database and stored in memory. The parameters are further categorized as parameter names and values. There are various parameters to each atomic filter. Examples of the parameter names are start pattern, end pattern, pattern to search, and pattern to apply. Not every parameter name is used in an atomic filter, and each filter may use different combinations of parameters.

At step 98 of Figure 3, the atomic filters are constructed by retrieving the atomic filter parameter names and values from the database. The parameters are constructed into an array and saved in memory. Retrieving the parameters from the database and constructing them when needed, guarantees that the atomic filters represent the current enabled filters selected by the user at the front-end of the system.

At step 100, the html (rich media) filters are constructed by retrieving the atomic filters, that were constructed in step 98, from memory. The atomic filters are constructed into an array and saved in memory as a combination

filter. If there are more enabled filters in the front end, step 101, then repeat steps 96, 98, 100, and 101. If there are no enabled filters, step 101, then construct a filter chain, step 102. At step 102, each of the combination filters constructed in step 100 are chained together so that each of them can be sequentially applied to the html file. At step 104, the filters are ready to be applied to the html file. The html file is retrieved from the database and stored in memory. At step 106, each combination filter is sequentially applied to the html file. The html file is searched for each atomic parameter value and marked with each corresponding atomic parameter name. The result is a list of marks within the html file that represent the atomic filter parameter names and values, which represent the combination filter. After the last atomic filter is marked within the html file, the combination filter is applied to the file. This results in the html file being modified to represent the filter choice made at the front-end of the system. This process continues until the last combination filter in the chain is applied. At step 108, if an error is detected, then the filtering process stops and a error string is created. If no errors are detected, then the results of step 106 are stored in the database, step 110.

Figure 4 illustrates the front-end of the pick-up URL process. The pick-up URL processes html files (images) retrieved from a web site and adds them to existing submissions. At step 120, the user enters a URL into the front-end system, using the browser, and the system automatically opens the web site of the particular URL. The web site is loaded into the browser so that the user can view it. At step 122, the user selects html files (creative assets)

from a list of html files that are displayed within the web site. At step 124, the selected files are saved into a file system and the list of html files are displayed. At step 126, the list of html files that the user selected in step 122 and a submission identification (id) code are sent to the back-end system for validating by the ISD/Validator system. The list of files and the id code are sent in the form of a string to the back-end, and a message is sent from the back-end to the front-end indicating that the string was properly received. After validating, a status message is sent by the back-end to the front-end which is displayed as either success or failure, step 128. When the user closes the automatic pick-up URL browser, it displays the submission with an updated list of assets (images).

Figure 5 illustrates the back-end of the pick-up URL process. Preliminary to step 130, the back-end receives the message sent by the front-end containing the string of html files and the id code. At step 130, the string is separated into separate strings so that each string is an individual file. At step 132, the corresponding e-mail information (html files) is updated in the database with the new files retrieved by the front-end pick-up URL process. The new html files are associated with other related html files sharing the same submission identification code and stored together in the database. Steps 134 and 136 are described in detail in figures 2 and 3 along with the supporting written description.

It will be appreciated that changes and modifications are likely to occur to those skilled in the art, and it is intended in the appended claims to cover all those changes and modifications which fall within the sprit and scope of the present invention.

**We claim:**

1. A method for processing submissions, using a computer, comprising the steps of:

a. receiving a submission;

b. searching the submission for a creative asset;

c. removing unnecessary code, attachments, or user selected code from the creative asset so that a modified creative asset exists in a form appropriate for use on a web page; and

d. comparing for conformity, properties of the modified creative asset to web site guidelines.

2. The method of claim 1, further comprising the step of displaying the modified creative asset for visual inspection.

3. The method of claim 1, further comprising the step of playing the modified creative asset for audible inspection.

4. The method of claim 1, further comprising the step of sending the modified creative asset to an ad server in preparation for distribution of the modified creative asset on a Web page.

5. The method of claim 1, wherein the submission may include multiple file attachments, html code, and information about where to locate a creative asset on the World Wide Web.

6. The method of claim 1, wherein the creative asset may be text code, image code, and rich media code.

7. The method of claim 6, wherein the rich media code may be JPEG, GIF, animated GIF, Enliven, InterVU, Comet Cursors, VRML, Shockwave, and Flash.

8. A system for processing submissions, on a computer system, comprising:

a front-end system for providing user control of the system;

a mail system for receiving, transmitting, and processing submissions into creative assets; and

a business system coupled to the front-end system and the mail system, wherein the business system receives creative assets and applies filters to creative assets, resulting in modified creative assets that are compared to ad models.

9. The system of claim 8, further comprising a database coupled to the computer for providing storage of rich media files, filters, and ad models.

10. The system of claim 9, further comprising an ad server database for storing submissions for use by a web site.

11. The system of claim 10, further comprising an adapter for interfacing the database with the ad server database.

12. The system of claim 8, wherein creative assets may be text code, image code, and rich media code.

13. The system of claim 12, wherein the rich media code may be JPEG, GIF, animated GIF, Enliven, InterVU, Comet Cursors, VRML, Shockwave, and Flash.

14. The system of claim 8, wherein the mail system further comprises a smtp system for receiving and transmitting e-mail.

15. The system of claim 8, wherein the mail system further comprises a POP system for receiving and transmitting e-mail.

16. The system of claim 8, wherein the mail system further comprises mail processors for separating the submission into components that may contain multiple file attachments, html code, or text that includes information about where to find creative assets on the World Wide Web.

17. The system of claim 8, wherein the filters remove unwanted code and add required code to the creative asset.

18. The system of claim 8, wherein the ad models are web site guidelines.

19. The system of claim 8, wherein the front-end system further comprises a pick-up URL system for locating submissions on the world wide web.

20. The system of claim 8, wherein the front-end system comprises at least one web browser system for viewing data and controlling data from a web server, where the web server provides the interface between the business system program and the web browser.

21. A computer implemented method for processing a submission received over a computer network comprising the steps of:

a. searching the submission for an attachment that contains a first rich media file;

b. creating a record for the first rich media file wherein the first rich media file is assigned an identification code;

c. parsing the first rich media file for a rich media file name referred within;

d. searching the submission for a second rich media file associated with the rich media file name identified during parsing;

e. storing the first rich media file and the second rich media file;

f. retrieving filter parameters from a database;

g. constructing filters;

h. applying the filters to the first rich media file so that a modified rich media file exists;

i. storing the modified rich media file;

j. assessing properties of the modified rich media file;

k. retrieving ad model parameters; and

l. checking the modified rich media file for compliance with ad model parameters.

22. The method of claim 21, wherein the rich media file is a hyper text markup language file.

23. The method of claim 21, further comprising before step (a), a step of receiving a submission.

24. The method of claim 21, wherein step (c) the first rich media file is searched for rich media file tags that reference image files.

25. The method of claim 24, wherein the rich media file tags are GIF, Animated GIF, JPEG, AU, AUI, InterVU, Comet Cursors, VRML, Class files, Flash, and Shockwave.

26. The method of claim 21, wherein step (g) further comprises the steps of combining filter parameters to form a filter.

27. A method of retrieving creative assets from a web site and adding them to existing submissions stored in a database comprising the steps of:

a. selecting a URL;

b. accessing the URL web site;

c. selecting creative assets from a list of files displayed at the web site;

d. storing the selected files in the database; and

e. attaching an identification code to the list of the selected files.

28. A computer system comprising:

a front-end system, wherein the front-end system provides a user with a window and controls through which one or more creative assets can be reviewed, and where the user can enable one or more filters and enable one or more ad models to apply to one or more creative assets; and

a back-end system, coupled to the front-end system, wherein the back-end system applies one or more filters to one or more creative assets resulting in one or more modified creative assets, and determines whether one or more modified creative assets conform with one or more ad models.

29. The computer system of claim 28, wherein the back-end system further comprises an email server, coupled to the Internet, for receiving one or more creative assets and for sending one or more modified creative assets to a third party.

30. The computer system of claim 28, wherein one or more creative assets are reviewed for content by the user.

31. The computer system of claim 28, wherein the back-end system, having Internet capabilities, locates one or more creative assets using a Uniform Resource Locator selected by the user and where the user selects one or more creative assets to be retrieved by the front-end system.

32. The computer system of claim 28, wherein the back-end system determines whether one or more modified creative assets conform with one or more ad models, by determining properties of one or more modified creative assets and verifying the properties against ad models.

33. The computer system of claim 32 wherein the ad models comprise of web site guidelines for displaying advertisements.

34. A system for processing a submission comprising:

a front-end system for providing an interface to the system for a user; and

a back-end system, coupled to the front-end system, for applying filters to the submission, determining properties of the submission, and applying ad model parameters to the submission to determine whether the submission complies with web site guidelines.

35. The system of claim 34 wherein the submission further comprises html code.

36. The system of claim 34 wherein the submission further comprises one or more attachments.

37. A computer implemented method for processing one or more creative assets comprising the steps of:

filtering one or more creative assets to produce one or more modified creative assets;

determining properties of one or more creative assets; and

comparing properties of one or more creative assets to one or more ad model parameters to determine compliance.

38. The method of claim 37 further comprising, before the step of filtering, the steps of:

retrieving filter parameters; and

constructing filters from the filter parameters.

39. The method of claim 37 further comprising, before the filtering step, the step of:

searching a submission for one or more creative asset; and

retrieving one or more creative assets referred to within the submission.

40. The method of claim 37 further comprising the step of reviewing one or more modified rich media files for content.

41. The method of claim 37 wherein the ad model parameters comprise of web site guidelines for displaying advertisements.

42. A computer implemented method for processing a submission comprising the steps of:

searching the submission for one or more html files;

parsing one or more html files for one or more creative assets referred to within the html file;

searching the submission for one or more creative assets referred to within one or more of the html files;

filtering one or more creative assets;

determining properties of one or more creative assets; and

determining whether one or more creative assets comply with one or more ad model parameters.

43. The method of claim 42 wherein the parsing step further comprises the step of searching the html file for tags that reference image files.

44. The method of claim 42 wherein the step of determining whether one or more creative assets comply with one or more ad model parameters

further comprises the step of comparing one or more creative asset properties to one or more ad model parameters.

45. The method of claim 42 further comprising the step of generating an acceptance status of one or more creative assets, if properties of one or more creative assets comply with ad model parameters.

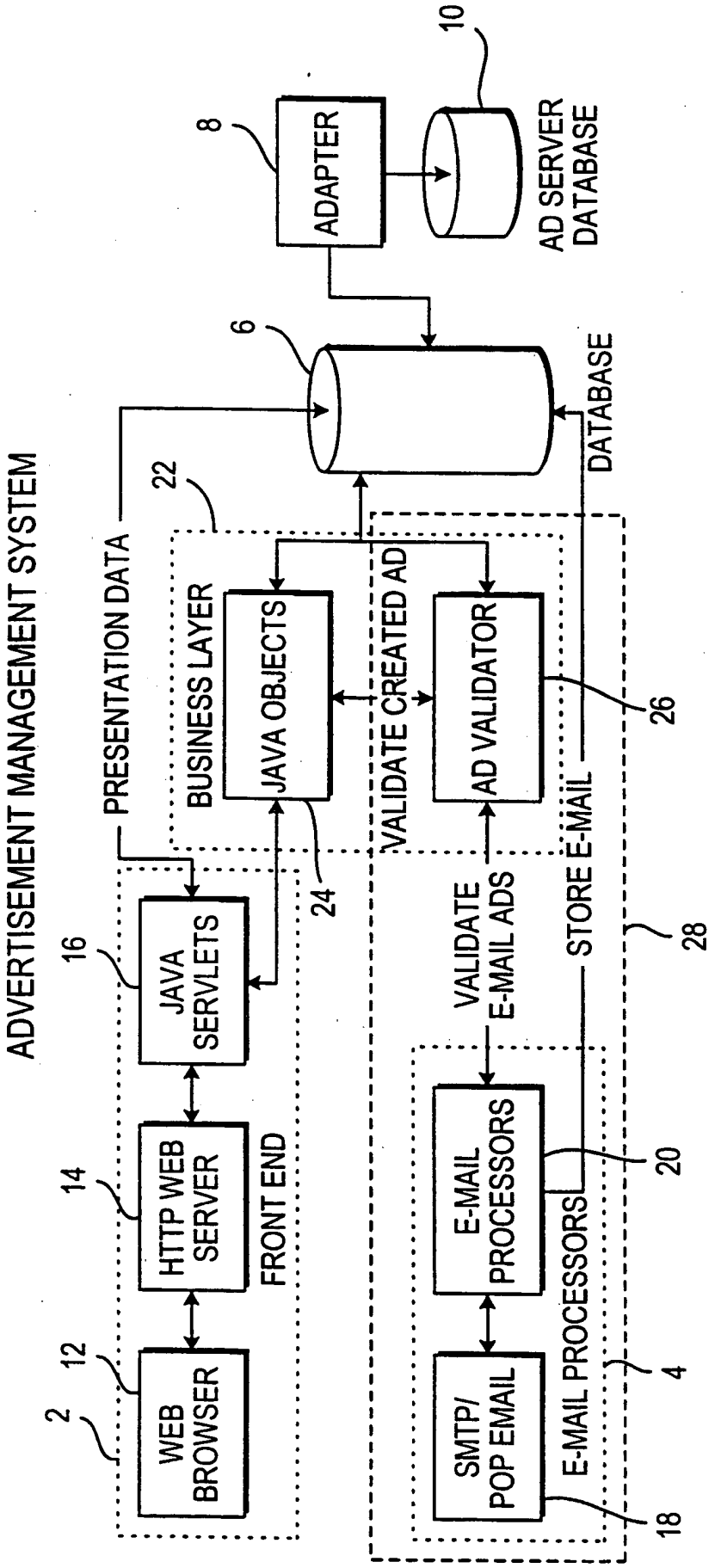

ADVERTISEMENT MANAGEMENT SYSTEM
ADAPTER
8
10
AD SERVER DATABASE
6
DATABASE
PRESENTATION DATA
22
BUSINESS LAYER
JAVA OBJECTS
24
VALIDATE CREATED AD
AD VALIDATOR
26
STORE E-MAIL
28
JAVA SERVLETS
16
HTTP WEB SERVER
14
FRONT END
WEB BROWSER
12
2
VALIDATE E-MAIL ADS
E-MAIL PROCESSORS
20
SMTP/ POP EMAIL
4
E-MAIL PROCESSORS
18

FIG. 1

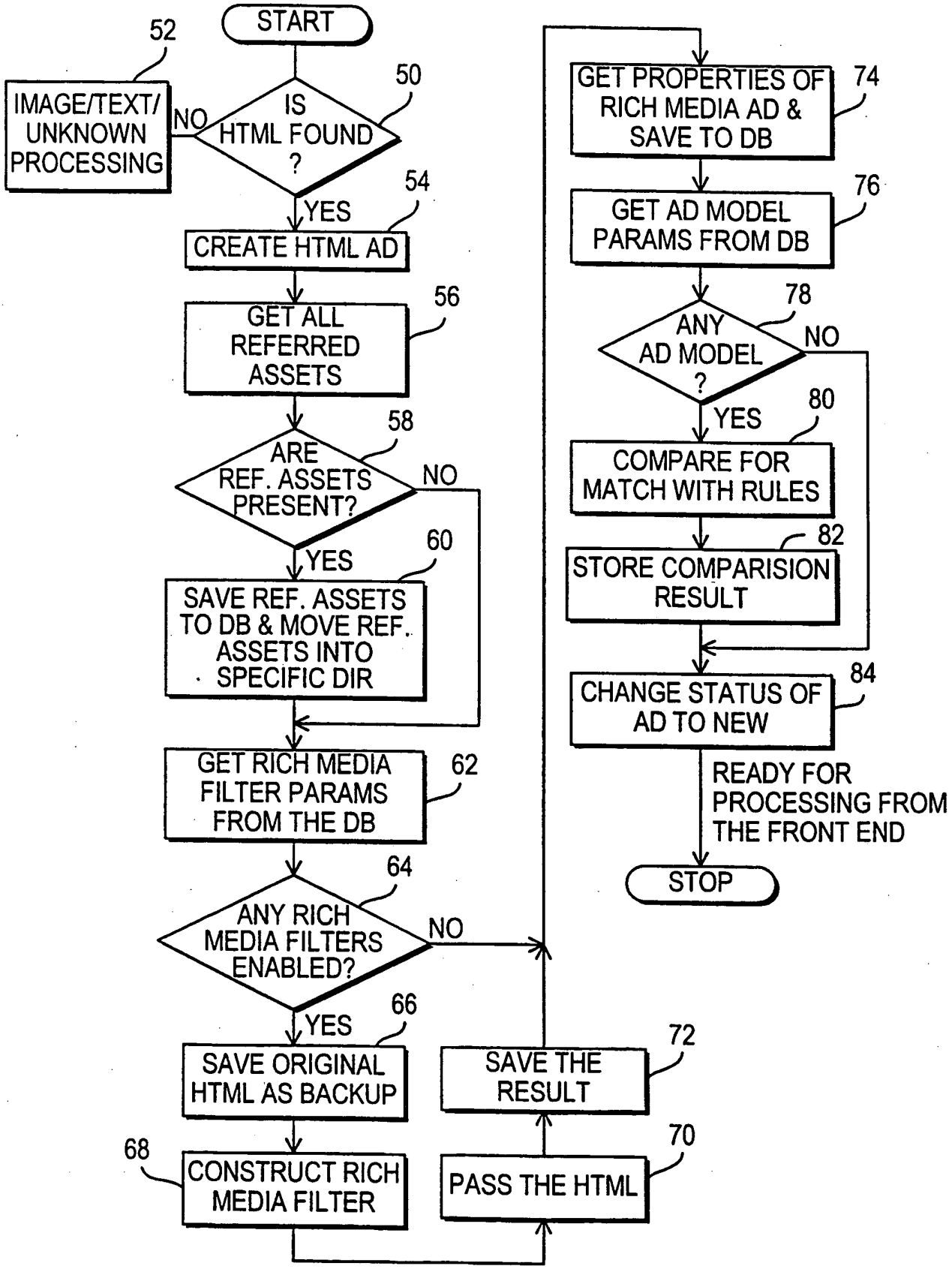

START
IS HTML FOUND ?
NO
IMAGE/TEXT/ UNKNOWN PROCESSING
YES
CREATE HTML AD
GET ALL REFERRED ASSETS
ARE REF. ASSETS PRESENT?
NO
YES
SAVE REF. ASSETS TO DB & MOVE REF. ASSETS INTO SPECIFIC DIR
GET RICH MEDIA FILTER PARAMS FROM THE DB
ANY RICH MEDIA FILTERS ENABLED?
NO
YES
SAVE ORIGINAL HTML AS BACKUP
CONSTRUCT RICH MEDIA FILTER
PASS THE HTML
SAVE THE RESULT
GET PROPERTIES OF RICH MEDIA AD & SAVE TO DB
GET AD MODEL PARAMS FROM DB
ANY AD MODEL ?
NO
YES
COMPARE FOR MATCH WITH RULES
STORE COMPARISION RESULT
CHANGE STATUS OF AD TO NEW
READY FOR PROCESSING FROM THE FRONT END
STOP
50
52
54
56
58
60
62
64
66
68
70
72
74
76
78
80
82
84

FIG. 2

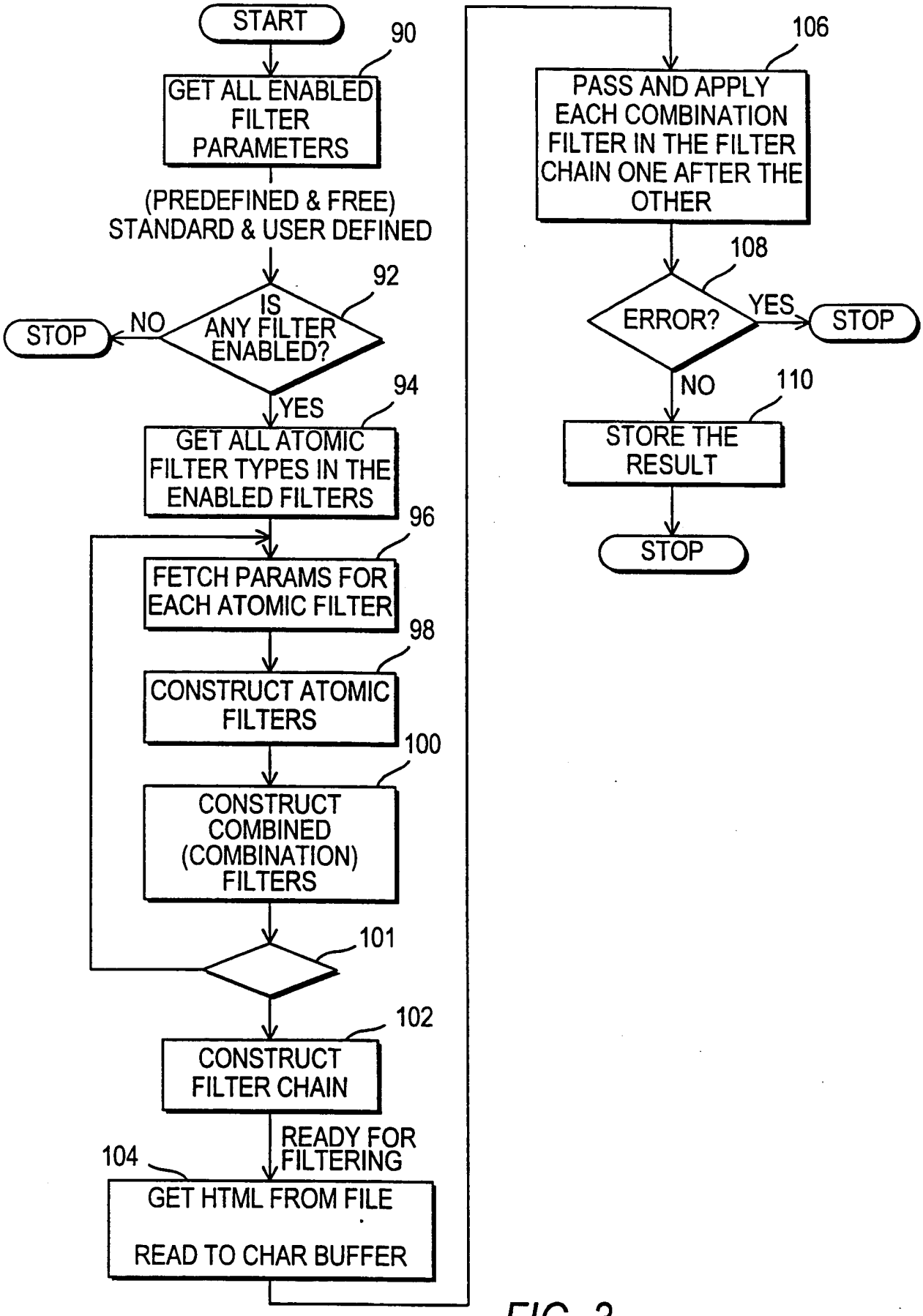
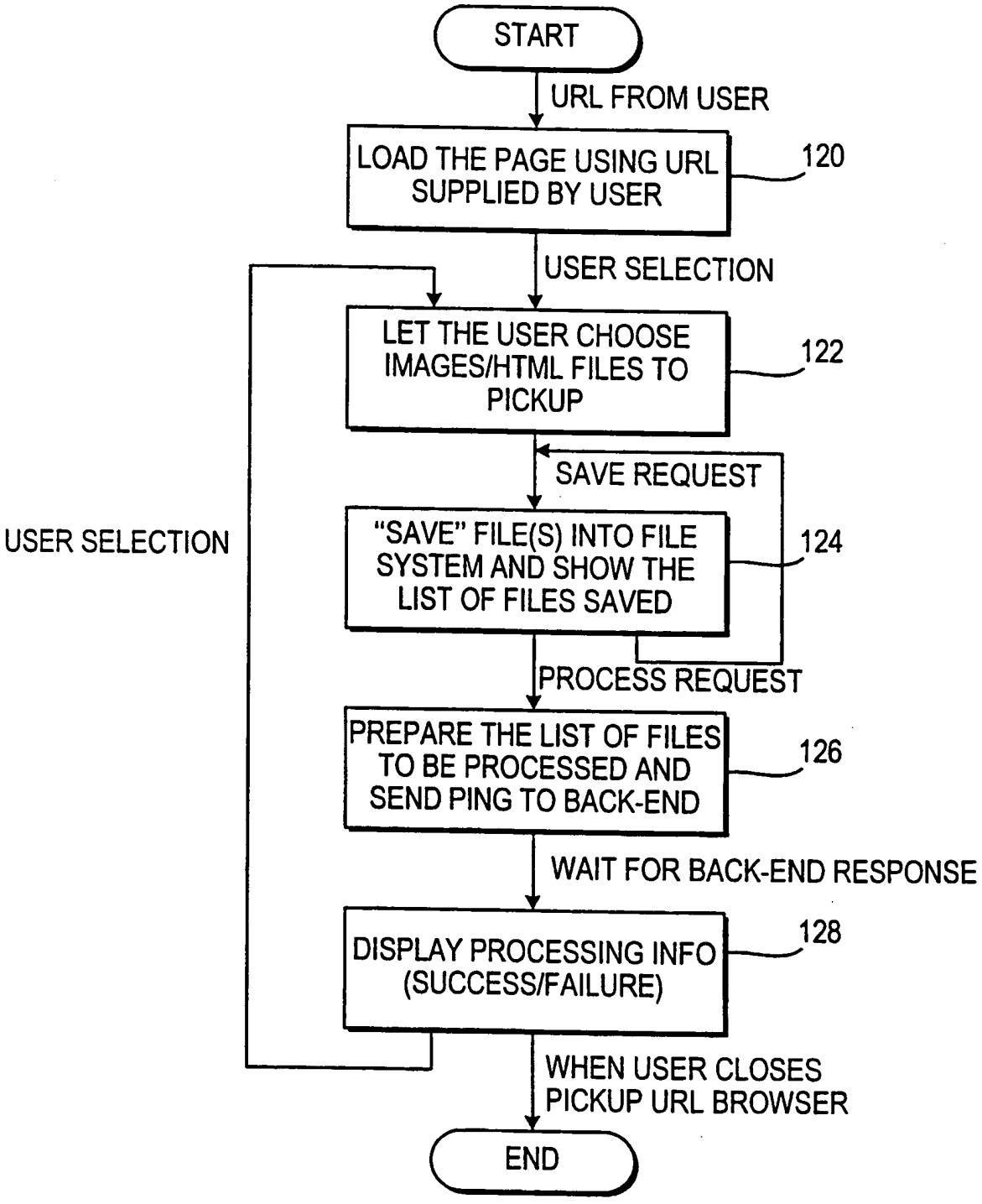
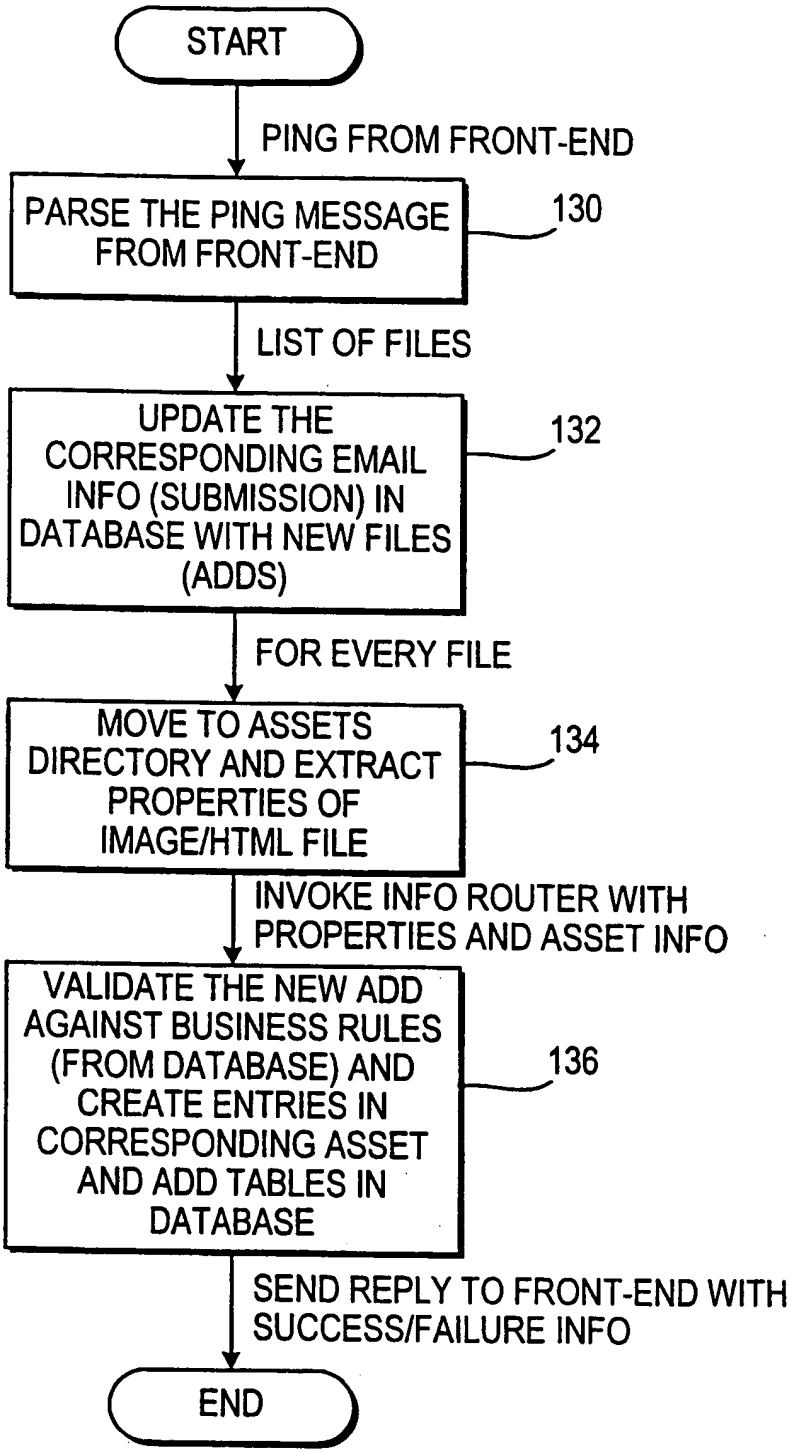

START
90
GET ALL ENABLED FILTER PARAMETERS
(PREDEFINED & FREE) STANDARD & USER DEFINED
92
IS ANY FILTER ENABLED?
NO
STOP
YES
94
GET ALL ATOMIC FILTER TYPES IN THE ENABLED FILTERS
96
FETCH PARAMS FOR EACH ATOMIC FILTER
98
CONSTRUCT ATOMIC FILTERS
100
CONSTRUCT COMBINED (COMBINATION) FILTERS
101
102
CONSTRUCT FILTER CHAIN
READY FOR FILTERING
104
GET HTML FROM FILE
READ TO CHAR BUFFER
106
PASS AND APPLY EACH COMBINATION FILTER IN THE FILTER CHAIN ONE AFTER THE OTHER
108
ERROR?
YES
STOP
NO
110
STORE THE RESULT
STOP

FIG. 3




PICKUP URL FLOW CHART: FRONT-END

FIG. 4




PICKUP URL FLOW CHART: BACKEND

*FIG. 5*